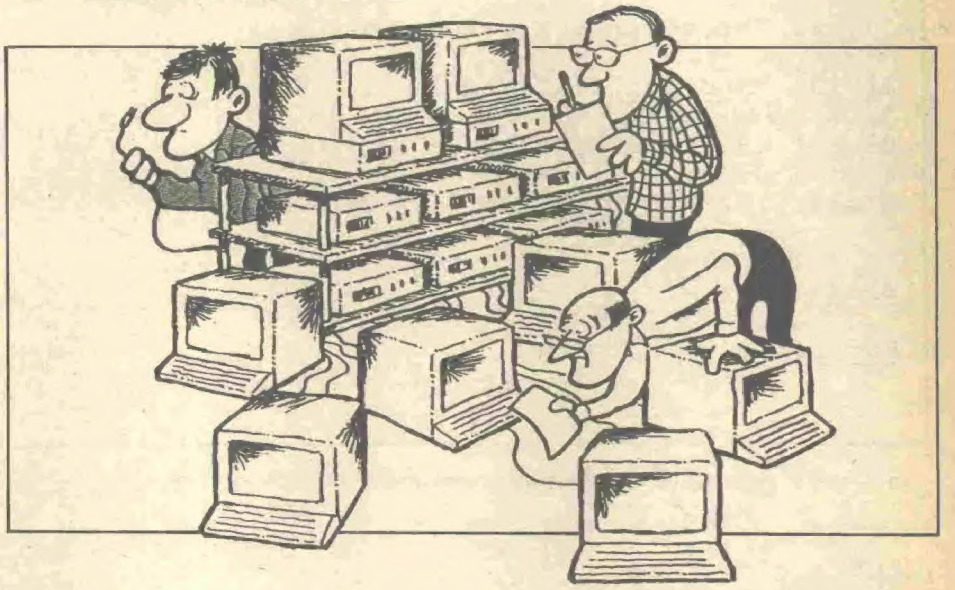
พรทิพย์ กิตติธนะศักดิ์

ตรวจสอบทางเลือกที่ใช้ Check box จำนวนมาก

แสดงให้คุณเห็นถึงวิธีการเขียนโค้ดอย่างง่าย ๆ สำหรับจัดการกับค่าเช็คบ็อกซ์จำนวนมาก



ผู้อ่านคงพบเห็นแอพพลิเคชันจำนวนมากบนวินโดวส์ที่มีการให้กำหนดทางเลือกว่าจะเลือกตัวเลือกใดบ้างโดยใช้เช็คบ็อกซ์ (Check box) หลาย ๆ ตัว ตัวอย่างเช่น คุณอาจจะต้องเซตเงื่อนไขจำนวนหนึ่งในการที่จะดึงข้อมูลขึ้นมาจากฐานข้อมูล หรือคุณอาจจะต้องออกแบบให้มีหลาย ๆ เงื่อนไขเพื่อการจัดรูปแบบของรายงานหนึ่ง ๆ

แอพพลิเคชันที่มีเช็คบ็อกซ์มากเท่าไรก็จะยิ่งทำให้มีทางให้เลือกมากขึ้น ยิ่งไปกว่านั้นจำนวนทางเลือกจะเพิ่มขึ้นสูงมากเมื่อเทียบกับการเพิ่มจำนวนของเช็คบ็อกซ์ เช่น ถ้ามีเช็คบ็อกซ์อยู่ 4 ตัว จะต้องเขียนโค้ดเพื่อจัดการกับทางเลือกที่มีถึง 16 ทางเลือก และถ้ามีเช็คบ็อกซ์ 6 ตัวก็จะมีได้ถึง 64 ทางเลือก ในบทความนี้เราจะแสดงให้คุณผู้อ่านเห็นถึงวิธีการเขียนโค้ดอย่างง่าย ๆ สำหรับจัดการกับค่าของเช็คบ็อกซ์เหล่านี้

ท้าวความ

กรณีที่แอพพลิเคชันหนึ่งมีการใช้เช็คบ็อกซ์กำหนดทางเลือกในการทำงานของโปรแกรม และถ้าการทำงานของฟังก์ชันใดฟังก์ชันหนึ่งเกิดจากผลของการเลือกเช็คบ็อกซ์เพียงตัวใดตัวหนึ่งเท่านั้น ไม่ได้ขึ้นกับเช็คบ็อกซ์ตัวอื่น ๆ เลย ผู้อ่านก็คงสามารถเขียนโปรแกรมจัดการให้ทำงานตามที่ต้องการได้โดยง่าย แต่ถ้าแอพพลิเคชันของเราสลับซับซ้อน การทำงานของแต่ละฟังก์ชันขึ้นกับความสัมพันธ์ของการเลือก (Select) เช็คบ็อกซ์จำนวนหนึ่งและการไม่เลือก (Deselect) เช็คบ็อกซ์อีกจำนวนหนึ่ง การเขียนโปรแกรมก็คงยุ่งยากมาก

ตัวอย่างเช่น แอพพลิเคชันหนึ่งมีฟังก์ชัน 4 ฟังก์ชันคือ ฟังก์ชัน A, B, C และ D มีเช็คบ็อกซ์สำหรับการตรวจสอบเงื่อนไขที่จะทำฟังก์ชันทั้งสี่นี้อยู่ 4 ตัว นั่นคือ เมื่อไรก็ตามที่เลือกเช็คบ็อกซ์ตัวที่ 1 หมายถึงต้องทำฟังก์ชัน A ถ้าไม่อย่างนั้นแล้วจะต้องเลือกเช็คบ็อกซ์ตัวที่ 2 หรือตัวที่ 3 ด้วย และถ้าเลือกเช็คบ็อกซ์ตัวที่ 2 และตัวที่ 3 ทั้งคู่ นั่นหมายถึงจะทำฟังก์ชัน B เสมอ ถ้าเลือกเช็คบ็อกซ์ตัวที่ 4 และต้องไม่เลือกเช็คบ็อกซ์ตัวที่ 1 จะหมายถึงต้องทำฟังก์ชัน C ส่วนกรณีอื่น ๆ ที่เหลือทั้งหมดให้ไปทำฟังก์ชัน D

จากตัวอย่างแอพพลิเคชันดังกล่าว คิดว่าควรจะเขียนโปรแกรมอย่างไรเพื่อจะตรวจสอบเงื่อนไขให้ถูกต้องได้วิธีอย่างง่าย ๆ วิธีหนึ่งก็คือ ใช้คำสั่ง IF/ELSE/ENDIF หลาย ๆ คำสั่งซ้อนกัน ประกอบกับการใช้โอเปอเรเตอร์ AND หรือ OR เข้าไปด้วย แต่การเลือกใช้วิธีนี้จะทำให้การเขียนโปรแกรมนั้นผิดพลาดได้ง่าย และการแก้ไขหาข้อผิดพลาดก็ทำได้อย่างยากลำบาก

```
' Turn the check box status into a logical value

B1 = (Check1.Value = 1)
B2 = (Check2.Value = 1)
B3 = (Check3.Value = 1)
B4 = (Check4.Value = 1)

' Test the check box values

If (B4) And Not (B1) Then
'     Perform C
ElseIf (B1) And Not (B2 or B3) Then
'     Perform A
ElseIf (B2 And B3) Then
'     Perform B
'     Perform C
Else
'     Perform D
End If
```

รูปที่ 1 แสดงโปรแกรมที่ใช้โครงสร้างที่ซับซ้อนของคำสั่ง IF/ELSE/ENDIF

ตัวอย่างโปรแกรมในรูปที่ 1 เป็นการใช้คำสั่ง IF/ELSE/ENDIF ในการทดสอบเงื่อนไขของแอพพลิเคชันตัวอย่าง

ถึงแม้ว่าโปรแกรมในรูปที่ 1 นั้นสามารถทำงานได้ถูกต้องดีก็ตาม แต่ปัญหาที่ตามมาก็คือ การปรับปรุงหรือบำรุงรักษาโปรแกรมนี้ในโอกาสข้างหน้าสำหรับผู้ที่ไม่ได้เป็นคนเขียนโปรแกรมนี้ตั้งแต่เริ่มแรกนั้นอาจจะทำได้ลำบากมาก

ทางเลือกใหม่ที่ดีกว่า

แทนที่เราจะต้องเขียนโปรแกรมโดยการใช้คำสั่ง IF/ELSE/ENDIF หลาย ๆ คำสั่งซับซ้อนกัน เราก็มีวิธีที่ง่ายกว่าคือ เริ่มจากการให้เช็คบ็อกซ์แต่ละตัวมีค่าเป็นเลขที่ไม่ซ้ำกัน คือ 1, 2, 4 และ 8 แล้วสร้างตัวแปรชั่วคราวขึ้นมาตัวหนึ่งชื่อว่า V และให้ค่าเริ่มต้นเป็นศูนย์ จากนั้นเมื่อมีการเลือกเช็คบ็อกซ์ตัวใด ๆ ก็จะเพิ่มค่าให้กับ V ด้วยค่าของเช็คบ็อกซ์ตัวนั้น ๆ ถ้าเลือกเช็คบ็อกซ์ตัวที่ 1 เราก็จะเพิ่มค่าให้กับ V ด้วยการบวก 1 เข้าไป ถ้าเลือกเช็คบ็อกซ์ตัวที่ 2 ก็บวกด้วย 2 เลือกเช็คบ็อกซ์ตัวที่ 3 บวกด้วย 4 และเมื่อเลือกเช็คบ็อกซ์ตัวที่ 4 ก็จะบวกด้วย 8

จะเห็นว่าวิธีคำนวณค่าของ V นั้นไม่ยาก สามารถเขียนโปรแกรมคำนวณที่เข้าใจได้ง่าย ๆ แต่ทว่าขั้นตอนที่เหลือถัดไปอาจจะยากขึ้นมาเล็กน้อย จากการเลือกหรือไม่เลือกเช็คบ็อกซ์แต่ละตัวนั้น เมื่อรวมค่าของเช็คบ็อกซ์ที่ถูกเลือกแต่ละตัวเข้าด้วยกันเป็นค่าของ V แล้วจะเห็นว่าค่าของ V ที่ได้ไม่ซ้ำกันเลย (Unique) ดังนั้นจึงสามารถใช้ค่านี้ในการกำหนดว่าจะให้ไปทำฟังก์ชันไหนได้อย่างง่าย ๆ

ข้อดีของวิธีนี้ก็คือ เราสามารถใช้คำสั่ง Select Case เพียงคำสั่งเดียวแทนการใช้คำสั่ง IF/ELSE/ENDIF ที่ซับซ้อนเข้าใจยาก ตัวอย่างโปรแกรมในรูปที่ 2 เป็นการแก้ปัญหาของแอพพลิเคชันในตัวอย่างข้างต้น ซึ่งใช้โปรแกรมเพียงไม่กี่บรรทัดก็สามารถจัดการได้

มาถึงตรงนี้ผู้อ่านบางท่านอาจจะแปลกใจว่า แล้วจะทำงานนี้ได้โดยวิธีง่าย ๆ ได้อย่างไร เราก็จะใช้ไดอะแกรมในตารางที่ 1 แสดงให้เห็น

ถ้าสังเกตค่าในตารางที่ 1 อย่างละเอียดก็จะเห็นว่า เครื่องหมายกากบาทในแต่ละแถวมีลักษณะรูปแบบเหมือนกับตัวเลขฐานสอง ซึ่งในตารางนี้มีค่าได้ตั้งแต่ 0 ถึง 15

เครื่องหมายกากบาทแต่ละตัวนั้นจะหมายถึงว่าเช็คบ็อกซ์นั้นถูกเลือก ค่าในคอลัมน์ "ผลรวม" ก็จะเป็นค่าที่ใช้ในคำสั่ง Select Case ส่วนคอลัมน์สุดท้ายเป็นฟังก์ชันที่ต้องทำสำหรับเงื่อนไขหนึ่ง ๆ

ถ้ามีความต้องการที่จะสร้างตารางของตนเองแล้ว เริ่มแรกต้องร่างหรือเขียนหมายเลขของเช็คบ็อกซ์ที่มีทั้งหมดไว้ที่บนสุดของตารางเสียก่อน จากนั้นคุณผู้อ่านก็เขียนค่าของเช็คบ็อกซ์แต่ละตัวไว้ใต้หมายเลขของมัน ถ้าผู้อ่านไม่คุ้นเคยกับระบบเลขฐานสองก็เพียงแต่ลอกแบบเครื่องหมายกากบาทจากตารางที่ 1 ลงในตารางที่สร้างขึ้นเองได้เลย ท้ายที่สุดจึงเขียนฟังก์ชันสำหรับเงื่อนไขที่เป็นไปได้ทั้งหมดลงที่คอลัมน์ "ฟังก์ชัน" เสร็จจากนี้แล้วก็พร้อมที่จะเขียนโค้ดในส่วนของคำสั่ง Select Case

วิธีการเขียนโค้ดในส่วนของคำสั่ง Select Case นี้ เริ่มต้นจากตรวจดูที่คอลัมน์ "ฟังก์ชัน" ว่ามีฟังก์ชันใดมากที่สุด ให้นำฟังก์ชันนั้นไปเขียนไว้ที่

```
Select Case V
      Case 1, 9
'           perform A
      Case 3, 7, 15
'           perform B
      Case 8, 10, 12
'           perform C
      Case 14
'           perform B
'           perform C
      Case else
'           perform D
End Select
```

รูปที่ 2 แสดงโปรแกรมที่ใช้โครงสร้างง่าย ๆ ของคำสั่ง Select Case แทนการใช้ดังในรูปที่ 1

Case Else การทำอย่างนี้จะช่วยให้เขียนโค้ดน้อยลงไปได้ เพราะว่าจะเหลือเงื่อนไขที่ต้องโค้ดน้อยลง จากในตัวอย่างจะพบว่ามีฟังก์ชัน D เกิดขึ้นถึง 7 ครั้ง ดังนั้นเราจึงเอาไปเขียนไว้ที่หลังคำ Case Else

จากนั้นให้ตรวจดูที่คอลัมน์ "ฟังก์ชัน" พร้อมไปกับคอลัมน์ "ผลรวม" ว่ามีค่าหรือเลขอะไรบ้างในคอลัมน์ "ผลรวม" ที่อยู่ในแถวเดียวกับฟังก์ชันนั้น แล้วให้เขียนค่าเหล่านั้นในคำสั่ง Select Case ที่แถวเดียวกันของ Case อันหนึ่ง ตัวอย่างเช่น ฟังก์ชัน A มีอยู่ที่เงื่อนไข 1 กับ 9 ดังนั้นวิธีเขียนคำสั่งที่ถูกจะเป็นดังนี้

```
Case 1, 9
'   perform A
```

ถ้าเขียนการคำนวณค่า V ไว้ในโพรซีเดอร์อีเวนท์คลิ๊ก (chick event) ของเช็คบ็อกซ์แต่ละตัวนั้น คุณจะต้องไม่ลืมว่าจะต้องมีการลบค่าออกจากค่า V เมื่อคุณยกเลิกการเลือกเช็คบ็อกซ์นั้น (deselect) วิธีคำนวณค่า V ที่ง่ายกว่าก็คือ รอให้เซตเช็คบ็อกซ์ทั้งหมดจนเสร็จก่อนแล้วค่อยคำนวณ

ตารางที่ 1 ถ้าให้ค่าในตารางนี้เป็นค่าของตัวแปร V แล้ว ก็สามารถที่จะใช้ V นี้ในการกำหนดว่าจะให้ทำฟังก์ชันใด

หมายเลขเช็คบ็อกซ์ :	4	3	2	1	ผลรวม	ฟังก์ชัน
ค่าของเช็คบ็อกซ์ :	8	4	2	1		
					0	D
				X	1	A
			X		2	D
			X	X	3	B
		X			4	D
		X		X	5	D
		X	X		6	D
		X	X	X	7	B
	X				8	C
	X			X	9	A
	X		X		10	C
	X		X	X	11	D
	X	X			12	C
	X	X		X	13	D
	X	X	X		14	BC
	X	X	X	X	15	B

X = เลือก (Selected) เช็คบ็อกซ์

บทสรุป

จากตัวอย่างแอพพลิเคชันที่ยกขึ้นมาแสดงนั้น เราใช้เช็คบ็อกซ์เพียง 4 ตัว ถ้าเราเก็บผลรวมค่าของเช็คบ็อกซ์ไว้ในตัวแปรชนิด Integer แล้ว สามารถที่จะใช้วิธีเขียนโปรแกรมแบบนี้เมื่อยังคงมีเช็คบ็อกซ์ไม่มากเกิน 15 ตัว แต่ถ้าเก็บในตัวแปรชนิด Long integer ก็จะสามารถใช้เช็คบ็อกซ์มากขึ้นไปอีก อย่างไรก็ตาม แม้ว่าวิธีที่ใช้อยู่นี้จะมีประสิทธิภาพเพียงใด ปัญหาที่อาจเกิดขึ้นได้จากการใช้วิธีนี้ก็คือ เมื่อมีการใช้เช็คบ็อกซ์มากเกินกว่าที่ตัวแปรจะรองรับได้